PATENT ABSTRACTS OF JAPAN

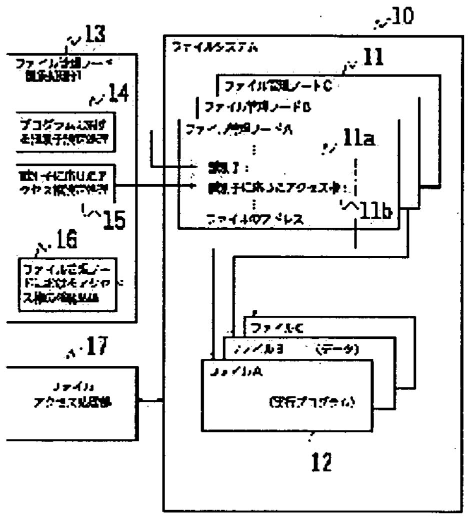
(11)Publication number : 05-100939
(43)Date of publication of application : 23.04.1993

(51)Int.Cl. G06F 12/00

(21)Application number : 03-213036
(22)Date of filing : 31.07.1991
(71)Applicant : FUJI XEROX CO LTD
(72)Inventor : HAYATA HIROSHI

(54) FILE SYSTEM

(57)Abstract:
PURPOSE: To execute read-out and write of a file only from a specific program by deciding an identifier of a program by an identifier of a file management node, and executing the access management by the access right corresponding to the identifier.
CONSTITUTION: An access right setting means 13 sets an identifier 11a given to a program of a file 12 as file management information to a file management node 11 for managing the file 12. Also, the access right 11b corresponding to the identifier 11a is registered and set as the access right of the file 12. In such a way, in the case of accessing the file 12 by executing the program, a file access managing means 17 decides an identifier of the program concerned by the identifier 11a set to the file management node 11. Subsequently, by this identifier, the access right 11b registered in the file management node 11 of the file 12 being an access object is discriminated. In accordance with information of this access right 11b, an access of the file 12 is controlled.



LEGAL STATUS
[Date of request for examination]
[Date of sending the examiner's decision of rejection]
[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]
[Date of final disposal for application]
[Patent number]
[Date of registration]
[Number of appeal against examiner's decision of rejection]
[Date of requesting appeal against examiner's decision of rejection]
[Date of extinction of right]

Japanese Patent Laid-open No. 5-100939

[Title of the Invention]

File System

[Abstract]

[Purpose] To provide a file system in which permission to read from a file and/or write to the file can be given only to a specific program.

[Constitution] A file system registers rights to access a file to a file management node associated with the file, and accesses the file according to the access rights registered to the file management node. An identifier to be given to a program of the file and access rights associated with the identifier are also registered to the file management node. When executing some program results in trying to access some file, the program's access to the file is managed in such a manner that the identifier registered to the file management node is determined as the identifier of the program, and the program is given the access rights which are set in combination with the identifier registered to the file management node associated with the file to be accessed.

[What is claimed is]

[Claim 1] A file system in which a right to access a file is registered to a file management node associated with the file and accesses to each file are performed according to the access right registered to the associated file management node, said system comprising:

access right setting means for registering an identifier to be given to a program contained in a file and a right to access a file associated with the identifier as a right to access the file to the file management node; and

file access management means for performing file access management in such a manner that when executing a program results in trying to access a file, said management means determines the identifier of the program by referring to the identifier registered to the file management node, and the program is given the access right which is set in combination with the identifier registered to the file management node associated with the file to be accessed.

[Detailed Description of the Invention]

[0001]

[Utilizing Field in Industry]

The present invention relates to a file system, and particularly to a file system for an information processing apparatus where the security of the system is raised by effectively using access right-based file management functions.

[0002]

[Prior Art]

Conventionally, in an information processing system, a collection of data having a certain purpose is treated as a file when data processing is done. Files increases explosively as the system grows in scale. For this reason, methods have been developed in order to treat many and various files in a systematic manner. For example, file management provides functions capable of managing a variety of files treated in an information processing system in a standard and systematic manner so that programs can perform file processing easily, efficiently and economically. These file management functions are provided as a file system, part of the operating system. A program performs file read and file write via an interface provided by the operating system for the file system. In this case, access right-based file management is done for each file so that such functions as data protection and system security protection can work.

[0003]

For example, the file system of the UNIX system uses a read system call to read data from a file while using a write system call to write data to a file (Refer to "THE DESIGN OF THE UNIX OPERATING SYSTEM , written by Maurie J Bach, translated by Fumi Sakamoto, Yoshinori Tada and Jun Murai, June 10, 1991, published by KYORITSU SHUPPAN

CO., Ltd., pp. 51-54, pp. 82-87 and other sections).

[0004]

In such a system, the control of read and write to a file in response to a file access request from a user is managed according to the rights to access the file. Information about the rights to access a file is registered to its i node (file management node) and is managed according to file management information. FIG. 6 is a schematic diagram indicating an example of the i node. This i node is composed of the following fields:

File owner identifier: Define individual and "group" owners who share and have rights to access the file.

File type: The file is any one of a regular file, a directory, a text file, block special file, and a FIFO (pipe).

File access permission: The system protects the file according to the three classes; the owner of the file, the group owner of the file, and other users. The rights to access the file for read (r), write (w) and execution (x) can respectively be set to each of the three classes. In the case of a directory, however, x means the class is permitted to search the directory for a filename because directories cannot be executed.

File access time: Indicates the time when the file was last updated, the time when the file was last accessed and the time when the i node was last accessed.

File data disk address table: Although the user treats the data in the file as a logical stream of bytes, the system kernel manages the data as scattered disk blocks. The i node identifies the disk blocks containing the file data.

File size: The address of each data in the file can be specified by its displacement counted up in bytes from the head of the file, that is, byte 0. The size of the file is larger by 1 than the largest byte displacement of data in the file. For example, if the user creates a file and writes one-byte data at byte displacement 1000, the file has a size of 1001 bytes.

[0005]

Shown in FIG. 6 is an example i node of a regular file owned by "MJB". This file contains 6030 bytes of data and 9-digit character data "rwxr-xr-x" is set as the permission modes (access rights). The first three characters "rwx" mean that the owner "MJB" is permitted by the file system to read from, write to and execute the file. Likewise, the subsequent three characters "r-x" mean that the members of a group named "OS" are permitted by the file system only to read from and execute the file while the last three characters "r-x" mean that the other members are permitted by the file system only to read from and execute the file. Thus, the members of the "OS" group and other users can read from and execute the file but cannot write to the file.

[0006]

Time information, such as last access time or last update time, is also held in i nodes for file management. In this example, the time information held in the i node indicates that the last read from a file was done by someone at 1:45 p.m., October 23, 1990 and the last write to the file was done by someone at 10:30 p.m., October 22, 1990.

[0007]

As described above, the file system of the UNIX system manages each file by providing a file management node (i node) to each file and setting file management information, such as rights to access the file or the owner of the file, to the file management node.

[0008]

[Problems to be Solved by the Invention]

Meanwhile, in the above file system, since each file is managed according to the file management information registered to the associated file management node, such as the rights to access the file and the owner of the file, any user can have a plurality of programs read from and write to the same file if the user acquires the access rights through some means. In the case of a database management system, for example, it is preferable that write to files is possible only from a specific program while regular programs are permitted only file read. If such a database management system is constructed based on the file management functions described above, trouble would occur in implementing such a specific program.

[0009]

The present invention has been made to solve the above mentioned problem and an object of the present invention is to provide a file system in which permission to read from a file and/or write to the file can be given to a specific program.

[0010]

[Means for Solving the Problems]

To achieve the above object, there is provided a file system according to the present invention in which a right to access a file (12; FIG. 1) is registered to a file management node (11; FIG. 1) associated with the file and accesses to each file are performed according to the access right registered to the associated file management node, characterized by comprising: access right setting means (13; FIG. 1) for registering an identifier to be given to a program contained in a file, and a right to access a file associated with the identifier as a right to the file to the file management node (11; FIG. 1); and file access management means (17; FIG. 1) for performing file access management in such a manner that when executing a program results in trying to access a file, the identifier registered to the file management node is determined as the identifier of the program, and the program is given the access rights which are set in combination with the identifier registered to the file management node associated with the file to be accessed.

[0011]

[Function]

In the file system, rights to access a file (12) are registered to a file management node (11) associated with the file. Rights to access each file are managed and access to each file is controlled according to the access rights registered to the associated file management node. This file system is provided with file access right setting means (13) and file access management means (17). To the file management node associated with each file, the access right setting means (13) sets an identifier to be given to the program contained in the file, and rights to access the file in combination with such identifiers. With this configuration, when executing a program results in trying to access a file, the file access management means (17) determines the identifier of the program by referring to the identifier registered in the file management node, determines the access right which is set in association with the identifier registered to the file management node of the file to be accessed, and controls the file access according to the information of the access right.

[0012]

As described above, when a file access request is issued from a file having an execution program in order to advance the execution of the program, the identifier of the program is determined and the file access is controlled in consistence with the access rights set for the identifier of the program. Thus, file access control based on access rights is possible at the level of execution programs in addition to the level of file owners and users, which brings about an increased degree of freedom in system construction including file operation, file processing and file management, and facilitates the implementation of a security-enhanced system configuration.

[0013]

[Embodiments]

A preferred embodiment of the present invention will be described hereinafter with reference to the drawings. FIG. 1 is a block diagram for explaining the key portions of a file system according to the embodiment of the invention. In FIG. 1, reference numeral 10 indicates a file system, 11 indicates a file management node, and 12 indicates a file. Files 12 are associated one-to-one with file management nodes 11. A file management node A is associated with a file A. A file management node B is associated with a file B. A file management node C is associated with a file C. The file management node 12 has an identifier field 11a for setting an identifier to the execution program contained in the file managed by this node, and an identifier-associated access right field 11b for setting access rights corresponding to the identifier.

[0014]

A file management node edit service block 13 is provided in order to individually register identifiers and identifier-dependent access rights to file management nodes 12 and check the file management information registered to these nodes. Service functions of this file management node edit service block 13 provide program identifier setting service 14, identifier-dependent access right setting service 15, file management node-examined access right check service 16, and so on.

[0015]

A file access service block 17 is also provided in the system in order to perform file access control for file access processing by using file management information registered to file management nodes.

[0016]

FIG. 2 is a diagram for explaining the relationships between files and their associated file management nodes with an example of file management information. FIG. 2(A) shows an example of a file management node associated with a data file while FIG. 2(B) shows an example of a file management node associated with an execution program file.

As with the file management node held in the conventional file systems, each of the file management nodes holds the file management information, such as the file owner, the file owner group, the last file access time, user-dependent access rights, or the actual disk address of the file. In this case, each of the file management nodes has a program identifier and program-dependent access rights registered thereto as file management information.

[0017]

A file management node 20 is associated with a file 21 containing data. This file management node registers, as file management information, an owner "Hayata", a group "FXKSP", last access time "Apr. 5 1991 19:00:00", last update time "Apr. 4 1991 12:30:00", user-dependent access rights "rwxr-xr-x", program-dependent access rights "(100 rwx)(101 r--)(102 r-x)", a program identifier "0", and a disk address "12345".

[0018]

A file management node 22 is associated with a file 23 containing an execution program. This file management node 22 registers, as file management information, an owner "Hayata", a group "FXKSP", last access time "Apr. 3 1991 19:00:00", last update time "Apr. 3 1991 12:30:00", user-dependent access rights "rwxr-xr-x", program-dependent access rights "0", a program identifier "100", and a disk address "22345".

[0019]

In this example, program-dependent access rights set to the data file A (21) as file management information are "(100 rwx)(101 r--)(102 r-x). This setting of program-dependent access rights means that a program given an identifier 100 is permitted read, write and execution, a program given an identifier 101 is permitted only read and a program given an identifier 102 is permitted read and execution but no write. Any other program is permitted none of read, write and execution. Note that "0" is set to the identifier field for the file A, that is, no program identifier is given because the file A is not an executable program file.

[0020]

An identifier "100" is set as file management information to the file B, an execution program file. This means the identifier 100 will be given to the program contained in the file B. Also note that no access rights are set to the identifier (program)-dependent access right field as file management information because the file B is not a data file.

[0021]

FIG. 3 is a flowchart illustrating an example of access right check service performed when a file is accessed using the file management information registered to the associated file management node. This processing is done by the file access service block (17; FIG. 1). First, the file management node associated with the execution program file is acquired in step 31. Then, in step 32, an ID is acquired as the identifier given to the program from the file management node. In step 33, the file management node associated with the file to be read is acquired. In step 34, program-dependent access right data A is read out from the file management node. In step 35, access right AC corresponding to the program identifier ID is acquired from the access right data A. In step 36, the contents of the access right AC are determined and access service according to the determined access right is performed. That is, if the access right AC includes read permission, return processing is done because the file can be read out and then execution returns to the main routine of the READ system call that is currently accessing the file. If the access right AC does not include read permission, since the file cannot be read, error return processing is done and file read error processing is performed.

[0022]

As described above, if a file is accessed by a program during execution of the program, an identifier given to the execution program is acquired from the associated file management node, access rights (access rights the corresponding program should have) set for this identifier are acquired from the file management node associated with the file to be accessed, and then file access is performed based on the access rights. Thus, this access right information-used access management provides file access control at the level of execution programs as well as at the level of file owners and users, and brings about an increased degree of freedom in system construction including file processing and file operation , and therefore facilitates the implementation of a security-enhanced system configuration.

[0023]

A description will be then made of the service functions which register file management information to file management nodes in this file system and check the information registered there. As described earlier, the file management node edit service block (13; FIG. 1) is provided in the embodiment in order to individually register identifiers, identifier-dependent access rights, and the like to file management nodes and check the file management information registered to these nodes. Respective service functions of this file management node edit service block provide program identifier setting service, identifier-dependent access right setting service, file management node-examined access right check service, and so on.

[0024]

FIG. 4 is a flowchart of program identifier setting service done to a file management node. FIG. 5 is a flowchart of program-dependent access right setting service done to a file management node. The program identifier setting service done to a file management node, shown in FIG. 4, acquires the associated file management node from the filename at first in step 41 and then sets a program identifier to this file management node in step 42. Practically, for example, the following function style program set_id is prepared as a procedure function to set program identifiers to the associated file management nodes:

set_id (filename, identifier)

set_id has two arguments; filename (execution program) and identifier. This is a procedure function to write a specified identifier to the file management node associated with a specified filename.

[0025]

The program-dependent access right setting service done to a file management node, shown in FIG. 5, acquires the associated file management node from the filename at first in step 51 and then sets identifiers in combination with access right data to this file management node in step 52. Practically, for example, the following function style program chapmod is prepared as a procedure function to set identifier (program)-dependent access rights to the associated file management nodes:

chapmod(filename, identifier, access right)
chapmod has three arguments; filename, identifier and access right. This is a procedure function to write information about a specified access right, which is to be applied to a specified identifier, to the file management node associated with a specified filename.

[0026]

In addition, an access right check function command which checks if each access is authorized by examining the access right applied to the identifier is implemented by using system interface functions provided for read and write file accesses, etc. That is, in the same manner as conventional user-dependent access right check service, access rights set to programs (identifiers) are checked.

[0027]

As described so far, in the file system according to the embodiment of the present invention, an identifier is given to each execution program file, an identifier is given to a program of the execution program file, and access rights are given to each data file in combination with identifiers. When a program tries to access a data file during execution, the identifier which has been given to the file containing the execution program will be used to determine the identifier given to the program, and on the basis of this identifier, the identifier-dependant access right of the data file is determined. File access control will be done on the basis of this determined access right. Thus, file management can be done at a program level in the same manner as at a user level. Further, it is possible to permit a file to be accessed only from a specific program if each program is given a unique identifier.

[0028]

[Effects of the Invention]

As described so far, according to the present invention, when a file is accessed by a program during execution of the program, an identifier given to the execution program is acquired from the associated file management node, access rights set for this identifier (access rights the corresponding program should have) are acquired from the file management node associated with the file to be accessed, and then file access is performed based on the access rights. Thus, this access right information-used access management provides file access control at the level of execution programs as well as at the level of file owners and users, and brings about an increased degree of freedom in system construction including file operation and file management, and therefore facilitates the implementation of a security-enhanced system configuration.

[Brief Description of the Drawings]

[FIG. 1]

FIG. 1 is a block diagram indicating key portions of a file system according to an embodiment of the present invention.

[FIG. 2]

FIG. 2 is a diagram for explaining relationships between file management nodes and respective files with an example of file management information data.

[FIG. 3]

FIG. 3 is a flowchart for illustrating an example of access right check service according to file management information registered to file management nodes when a file is accessed.

[FIG. 4]

FIG. 4 is a flowchart for illustrating program identifier setting service done to a file management node.

[FIG. 5]

FIG. 5 is a flowchart for illustrating program-dependent access right setting service done to a file management node.

[FIG. 6]

FIG. 6 is a diagram for explaining an example of i node, a file management node.

[Description of the Symbols]

10. File system

11. File management node

11a. Identifier field

11b. Identifier and access right field

12. File

13. File management node edit service block

17. File access service block

20. File management node A

21. File A (Data file)

22. File management node B

21. File B (Execution program file)

In the drawings:

FIG. 1

10: File system

11: File management node C

11B: File management node B

11A: File management node A

11a: Identifier

11b: Identifier-dependent access right

100: File address

12C: File C

12B: File B (Data)

12: File A (Execution program)

13: File management node edit service block

14: Program identifier setting service

15: Identifier-dependent access right setting service

16: File management node-examined access right check service

17: File access service block

FIG. 2 (A)

20: File management node A

200: Owner

201: Group

202: Last access time

203: Last update time

204: User-dependent access right

205: Program-dependent access right

206: Identifier given to program

207: Disk address

21: File A (Data)

FIG. 2 (B)

22: File management node B

221: Owner

222: Group

223: Last access time

224: Last update time

225: User-dependent access right

226: Program-dependent access right

227: Identifier given to program

228: Disk address

23: File B (Execution program)

FIG. 4

400: Program identifier setting service

41: Acquires a file management node associated with the filename.

42: Sets a program identifier to the file management node.

FIG. 5

500: Access right setting service

51: Acquires a file management node associated with the filename.

52: Sets identifiers and corresponding access rights to the file management node.

FIG. 3

300: Access right check service

31: Acquires a file management node associated with the execution program file.

32: Acquires identifier ID given to the program from the file management node.

33: Acquires a file management node associated with the file to be read.

34: Read out program-dependent access right A from the file management node.

35: Acquires access right AC for program identifier ID from the access right data A.

36: AC includes read permission?

37: Read error processing

FIG. 6

600: i node

610: Owner

620: Group

630: File type: regular file type

640: Permission mode

650: Last access time

660: Last update time

670: i node last update time

680: Size: 6030 bytes

690: Disk address

報を用いてファイルアクセス時に行なわれるアクセス権チェック処理の一例を示すフローチャート、

【図4】　図4はファイル管理ノードに対するプログラム識別子設定処理を示すフローチャート、

【図5】　図5はファイル管理ノードに対するプログラム対応のアクセス権

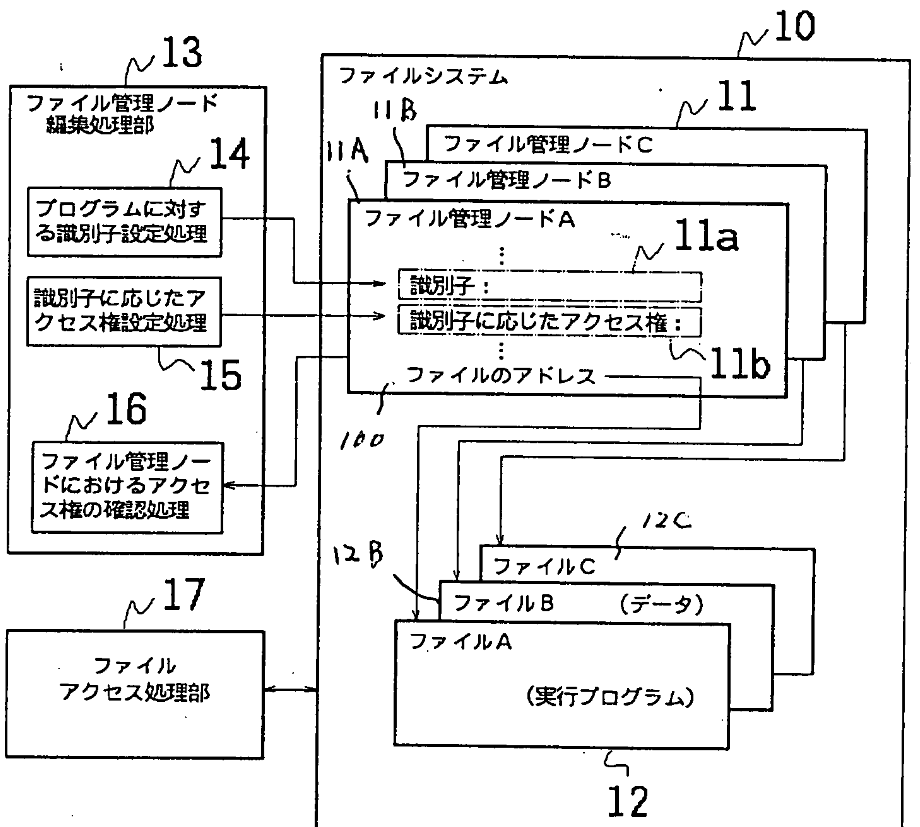
【図6】　図6はファイル管理ノードであるｉノードの一例を説明する図である。

【符号の説明】

１０…ファイルシステム、１１…ファイル管理ノード、１１ａ…識別子フィールド、１１ｂ…識別子アクセス権フィールド、１２…ファイル、１３…ファイル管理ノード編集処理部、１７…ファイルアクセス処理部、２０…ファイル管理ノードＡ、２１…ファイルＡ（データファイル）、２２…ファイル管理ノードＢ、２１…ファイルＢ（実行プログラムファイル）。

【図1】

図1



【図2】

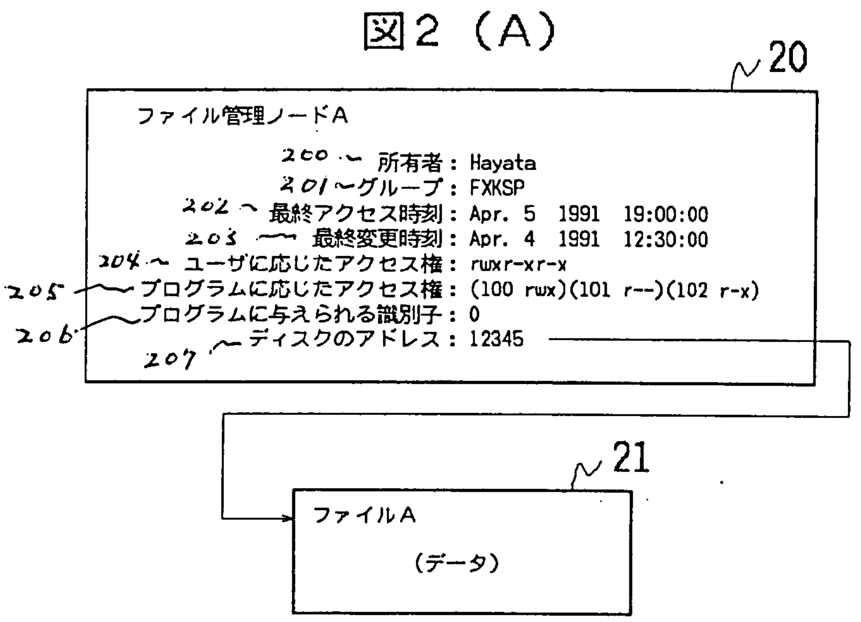


図2（A）
20
ファイル管理ノードA
200 所有者：Hayata
201 グループ：FXKSP
202 最終アクセス時刻：Apr. 5 1991 19:00:00
203 最終変更時刻：Apr. 4 1991 12:30:00
204 ユーザに応じたアクセス権：rwxr-xr-x
205 プログラムに応じたアクセス権：(100 rwx)(101 r--)(102 r-x)
206 プログラムに与えられる識別子：0
207 ディスクのアドレス：12345
21
ファイルA
（データ）

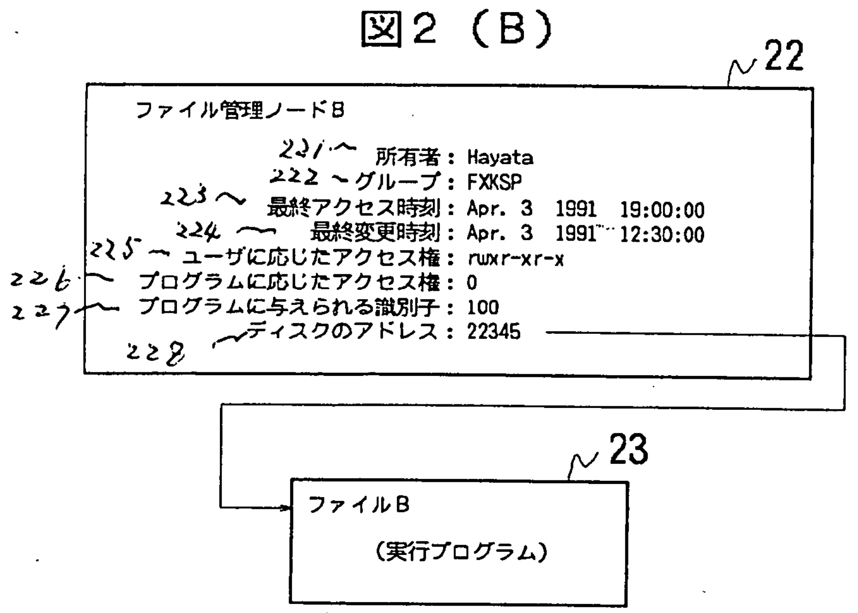


図2（B）
22
ファイル管理ノードB
221 所有者：Hayata
222 グループ：FXKSP
223 最終アクセス時刻：Apr. 3 1991 19:00:00
224 最終変更時刻：Apr. 3 1991 12:30:00
225 ユーザに応じたアクセス権：rwxr-xr-x
226 プログラムに応じたアクセス権：0
227 プログラムに与えられる識別子：100
228 ディスクのアドレス：22345
23
ファイルB
（実行プログラム）

【図4】

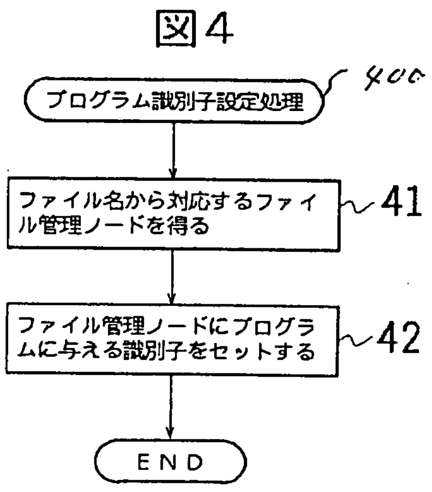


図4
プログラム識別子設定処理
400
ファイル名から対応するファイル管理ノードを得る
41
ファイル管理ノードにプログラムに与える識別子をセットする
42
END

【図5】

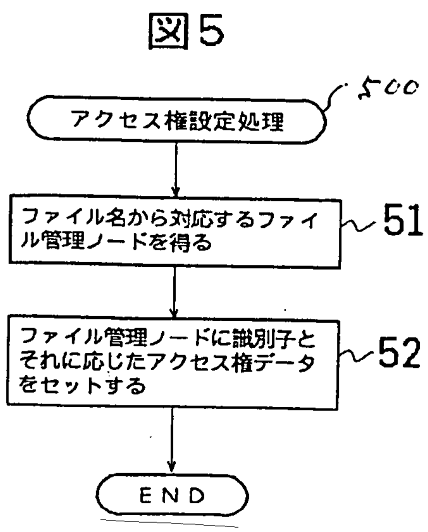


図5
アクセス権設定処理
500
ファイル名から対応するファイル管理ノードを得る
51
ファイル管理ノードに識別子とそれに応じたアクセス権データをセットする
52
END

【図3】

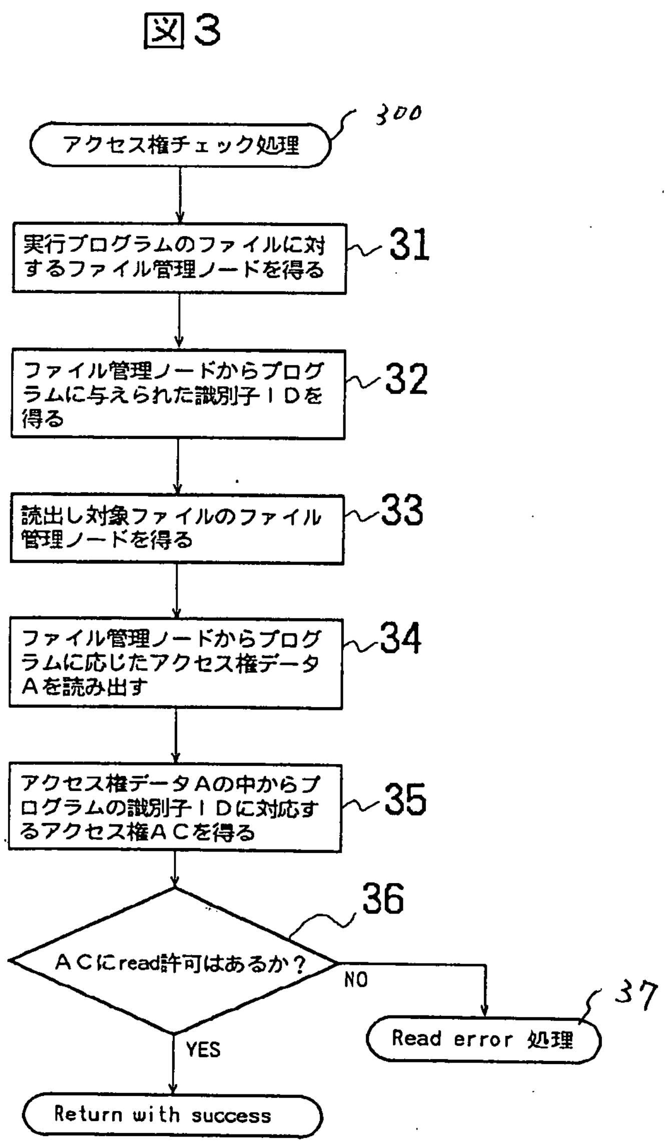


図3
300
アクセス権チェック処理
実行プログラムのファイルに対するファイル管理ノードを得る
31
ファイル管理ノードからプログラムに与えられた識別子IDを得る
32
読出し対象ファイルのファイル管理ノードを得る
33
ファイル管理ノードからプログラムに応じたアクセス権データAを読み出す
34
アクセス権データAの中からプログラムの識別子IDに対応するアクセス権ACを得る
35
ACにread許可はあるか？
36
NO
Read error 処理
37
YES
Return with success